Concurrent Programming in Cocoa

Session 108

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Concurrent Programming in Cocoa

Chris Kane
Cocoa Frameworks

To Be Covered…

- Review NSOperation, NSOperationQueue
- What's New with NSOperations?
- New Collection API
- New NSNotificationCenter API
- Concurrent Document Opening
- Concurrent Drawing

Operations and Queues

Operations and Queues

- Cornerstone of concurrency
  - In Cocoa: NSOperation and NSOperationQueue
- Replacing threads as principle concurrency mechanism
- Snow Leopard introduces two major features at lower levels
  - Blocks
  - Grand Central Dispatch (GCD)
- NSOperationQueue re-implemented atop GCD

Let’s Review…

NSOperation

- An object to encapsulate and represent “work to do”
- Also encapsulates various properties
  - isExecuting, isFinished
  - Cancellation
  - Queue priority
  - Dependencies

Defining NSOperations

- Use NSBlockOperation or NSInvocationOperation
- Or subclass NSOperation and override -main

Running NSOperations

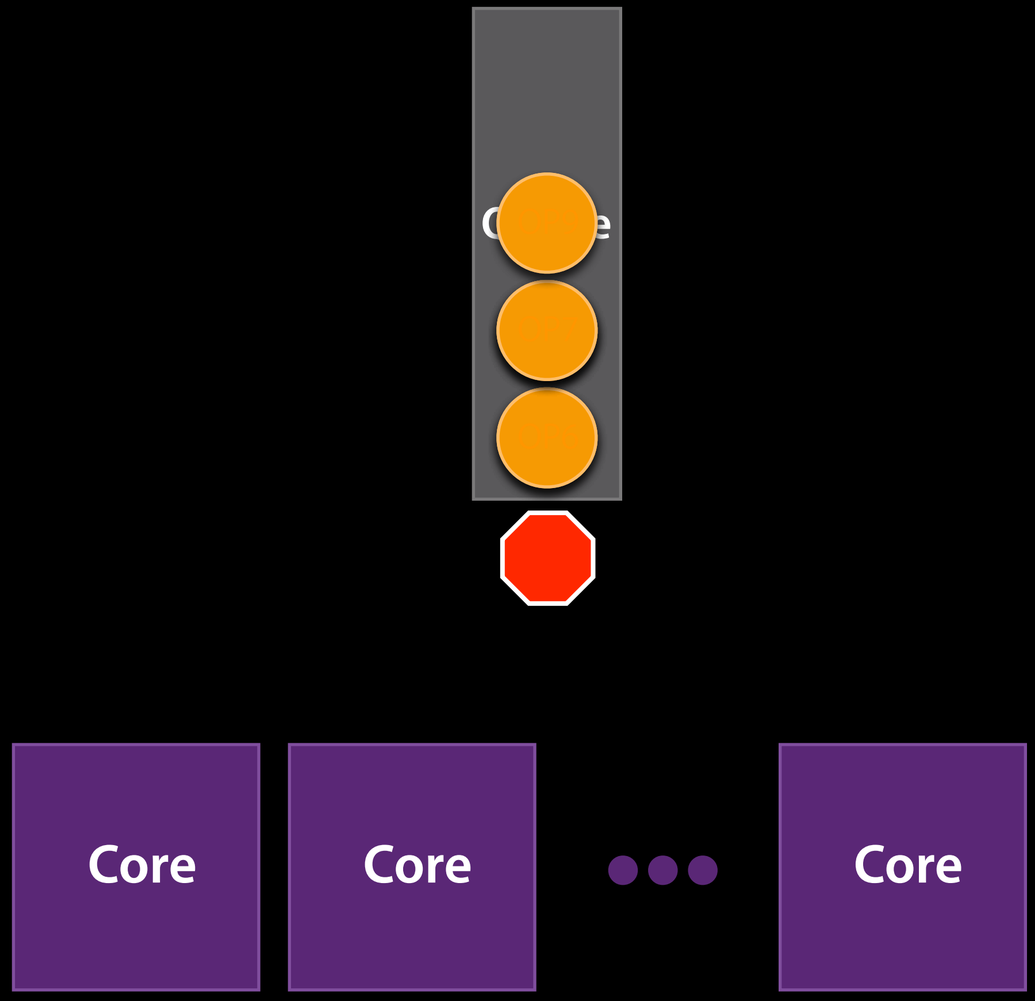
- Can execute operations directly
- But usually put them in an NSOperationQueue

NSOperationQueue

- Mechanism for concurrently executing NSOperations
  - Can also execute operations serially
- You can
  - Add operations
  - Get the list of operations
  - Adjust the “width” of the queue
  - Suspend and resume a queue
  - Wait for all operations to finish

NSOperationQueue Illustrated

- Add operations
- Suspend and resume a queue
- Wait for all operations to finish



NSOperation and Queues in 10.6

What's New with NSOperations?

- New NSOperation API
  - waitUntilFinished method
  - completionBlock property
- Completion block is run after the operation finishes
  - Block is not run in any particular execution context
  - Execution is concurrent with dependencies

NSBlockOperation

- New subclass of NSOperation
  - Holds one or more blocks
- The blocks are executed concurrently
- The blocks have no parameters and return no result

```
void (^)(void)
```

Return type

Parameter types

Objective-C and Garbage Collection Advancements	Russian Hill Wednesday 5:00PM

What's New with NSOperationQueues?

- New NSOperationQueue API
  - addOperationWithBlock: method
  - addOperations:waitUntilFinished: method
  - operationCount property

Operation Queues and GCD

- NSOperationQueue re-implemented atop GCD
- There is no 1:1 correspondence between dispatch queues and operation queues
  - NSOperations are run (started) on a concurrent dispatch queue
- Exception is NSOperationQueue's and GCD's main queues

NSOperationQueue +mainQueue

- +mainQueue returns the queue bound to the main thread
  - Operations run on the main thread
  - This is a serial queue
  - Its properties can’t be modified
  - Corresponds to and uses the GCD main queue

The Main Queues

- In Cocoa apps, the main run loop services the main queues
- The main queues act like a run loop source which has been put in the “common modes”
  - Serviced only when the run loop is running in one of the common modes
  - Not serviced re-entrantly, same as other run loop sources
- A new way to get work “back to the main thread”
  - Similar in effect to async -performSelectorOnMainThread: …

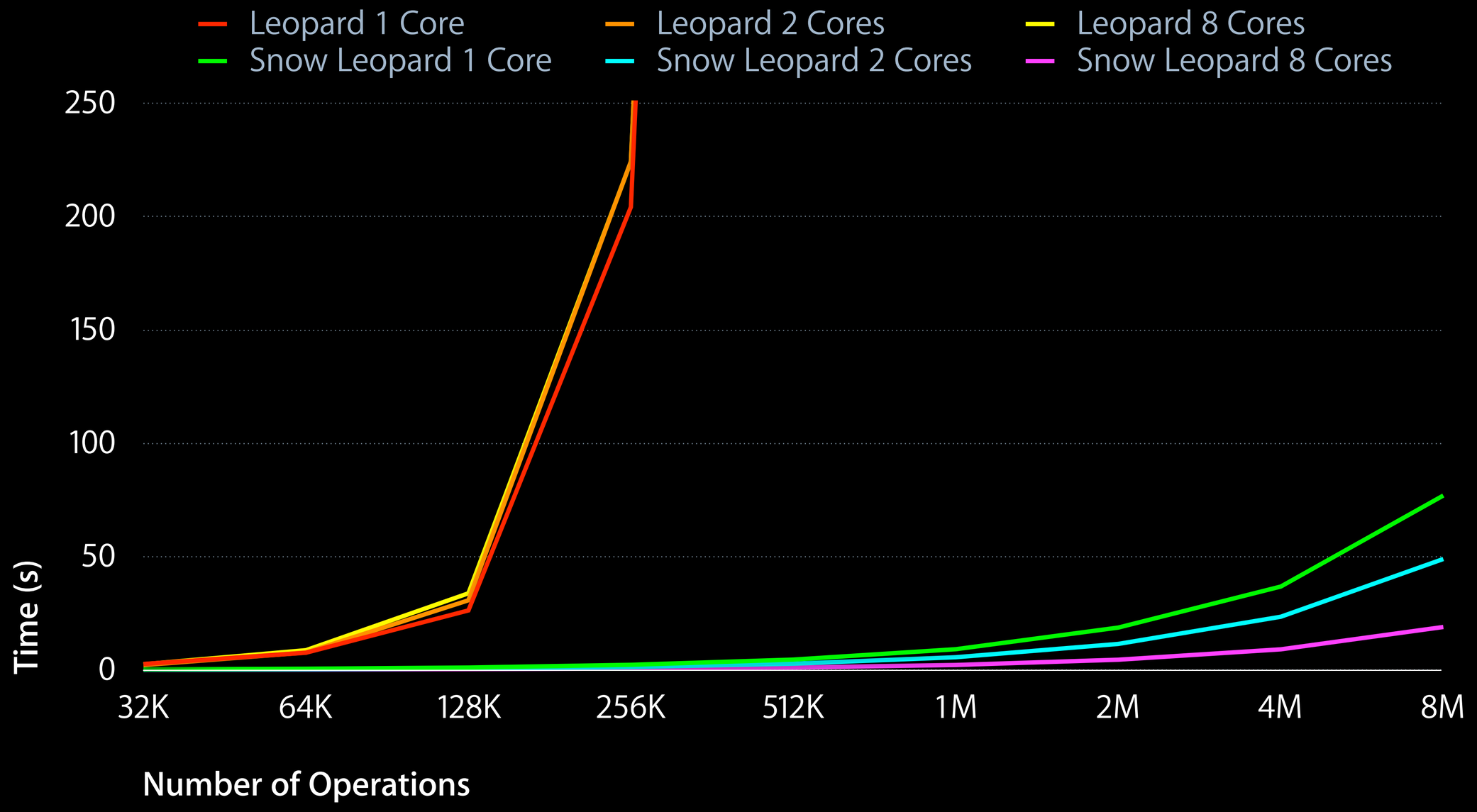
NSOperationQueue +currentQueue

- +currentQueue returns the queue running the executing operation
  - This method will usually return nil in other contexts
  - Always returns the mainQueue on the main thread
- Can be tricky to use
  - If it isn't your queue, interesting things can happen
  - If it isn't your queue, you should not modify its properties

Better Performance

- Improved performance over Leopard
- Faster adding of operations to queues
  - About 2.8x faster on multicore Intel machines
- Greater asynchrony, simplified concurrency safety
- Lower execution overhead per operation
  - Overhead: Amount of time to execute a do-nothing operation

Operation Overhead



Collections

New Collection APIs

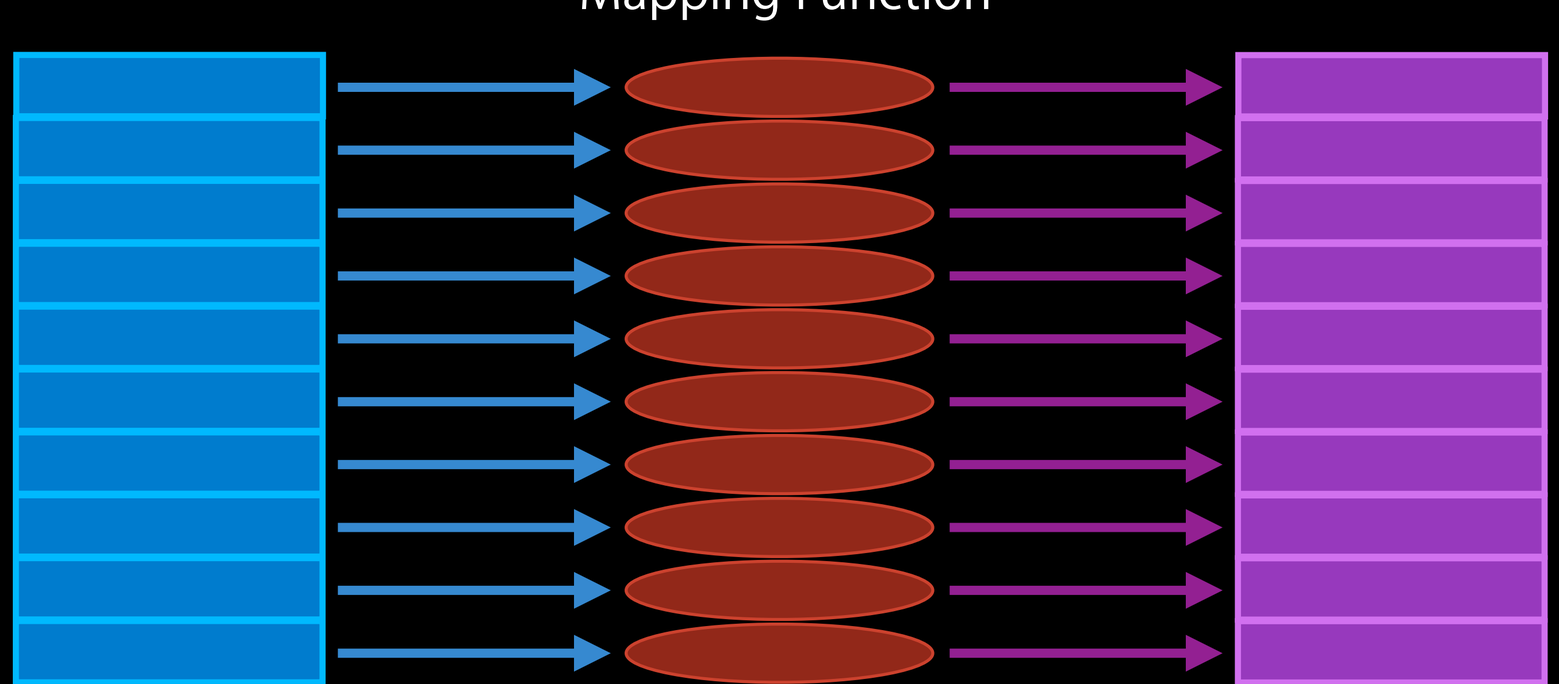
- New methods on NSArray, NSDictionary, NSSet, NSIndexSet
  - Enumeration and searching
- -enumerateObjectsWithOptions:usingBlock:

```
void (^)(id obj, NSUInteger index, BOOL *stop)
```

- Option: NSEnumerationConcurrent
  - Block may be invoked concurrently

Using Concurrent Enumeration

- Implement “map” function on NSArray
  - Use the elements of one array to produce a corresponding array



Using Concurrent Enumeration

- Design points for "map"
  - Caller will specify a block as the mapping function
  - Block will get object and index
  - No "stop" option for the block
  - Block returns object (autoreleased) for new array, and not nil
  - Block will be invoked concurrently, must be safe
  - Map method will return new NSArray

Using Concurrent Enumeration

```
- (NSArray *)my_mapUsingBlock:(id (^)(id obj, NSUInteger idx))mapper {
    NSUInteger cnt = [self count];
    id *temp = (id *)malloc(cnt * sizeof(id));
```

- ‘temp’ is a fixed-length temporary C array
  - Don’t have to keep track of ordering
  - Don’t have to thread-safely access some collection

Using Concurrent Enumeration

```
- (NSArray *)my_mapUsingBlock:(id (^)(id obj, NSUInteger idx))mapper {
    NSUInteger cnt = [self count];
    id *temp = (id *)malloc(cnt * sizeof(id));
    [self enumerateObjectsWithOptions: NSEnumerationConcurrent usingBlock:
        ^(id obj, NSUInteger idx, BOOL *stop) {
            temp[idx] = [ mapper(obj, idx) retain ];
    }];
```

Using Concurrent Enumeration

```
- (NSArray *)my_mapUsingBlock:(id (^)(id obj, NSUInteger idx))mapper {
    NSUInteger cnt = [self count];
    id *temp = (id *)malloc(cnt * sizeof(id));
    [self enumerateObjectsWithOptions: NSEnumerationConcurrent usingBlock:
        ^(id obj, NSUInteger idx, BOOL *stop) {
            temp[idx] = [ mapper(obj, idx) retain ];
        }];
    NSArray *result = [NSArray arrayWithObjects: temp count: cnt];
    for (NSUInteger idx = 0; idx < cnt; idx++) [temp[idx] release];
    free(temp);
    return result;
}
```

Concurrent Sorting

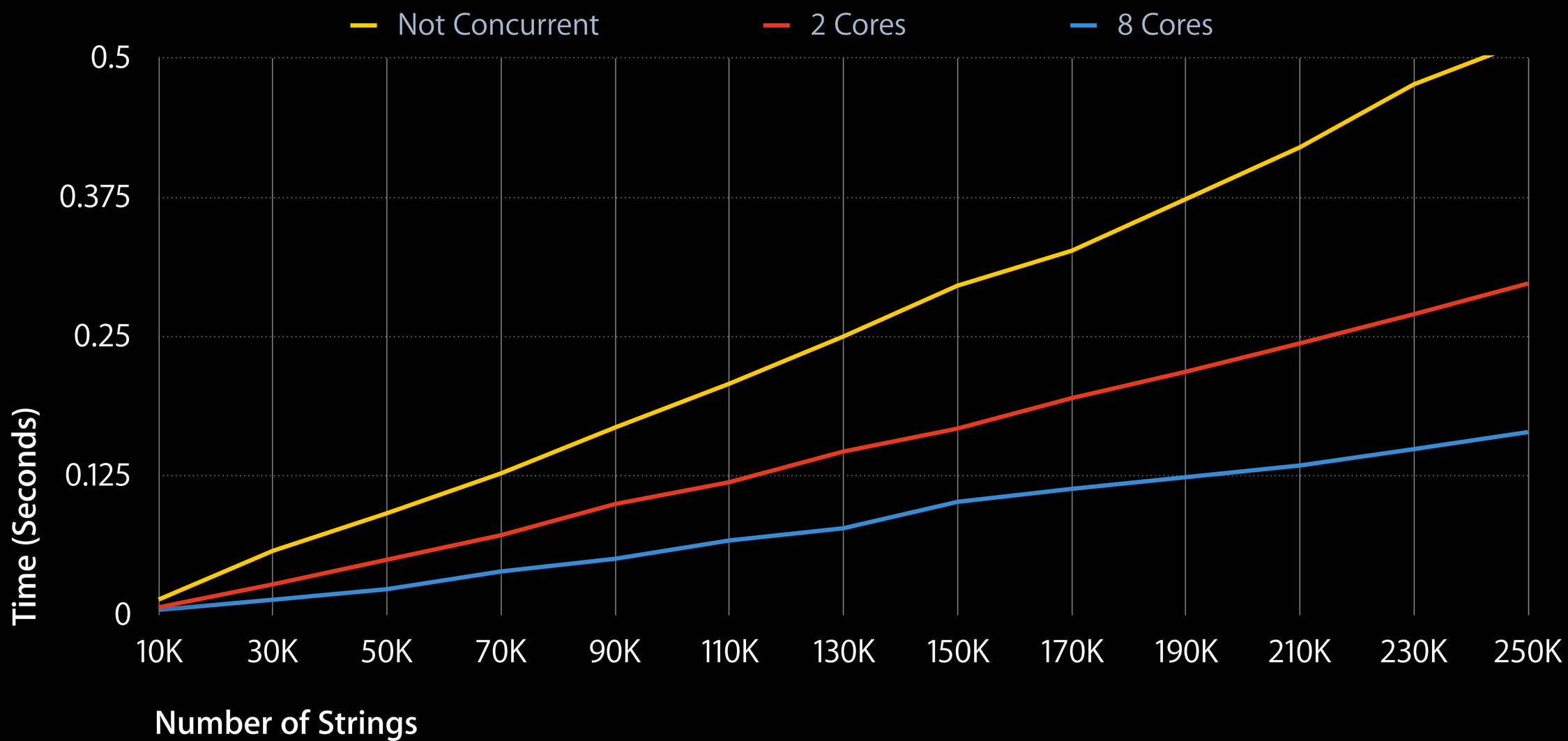
- New methods on NSArray, NSMutableArray
- -sortedArrayWithOptions:… usingComparator:(NSComparator)block
  - Use NSSortConcurrent option for concurrent sorting
- NSComparator block type

```
NSComparisonResult (^)(id obj1, id obj2)
```

- Your comparator block must be concurrency-safe
- NSSortStable option can be used with NSSortConcurrent

Concurrent Sorting

Array of random short strings



NSNotificationCenter

NSNotificationCenter

```
- (id)addObserverForName:(NSString *)name
      object:(id)obj
      queue:(NSOperationQueue *)queue
      usingBlock:(void (^)(NSNotification *))block
```

- New observer-adding API
- Traditional method takes observer + selector, returns void
- Block will be invoked when notification posted

NSNotificationCenter

```
- (id)addObserverForName:(NSString *)name
      object:(id)obj
      queue:(NSOperationQueue *)queue
      usingBlock:(void (^)(NSNotification *))block
```

- Block executes in the context of the given queue
- Observer blocks are executed concurrently when possible
  - Posting still synchronous
- If queue is nil, executes in the posting thread

NSNotificationCenter

```
- (id)addObserverForName:(NSString *)name
      object:(id)obj
      queue:(NSOperationQueue *)queue
      usingBlock:(void (^)(NSNotification *))block
```

- Returned object must be retained and held
- Use -removeObserver: With this object to cancel registration

Concurrent Document Opening

NSDocument

- NSDocument can now open files concurrently
  - TextEdit does this for most document types
  - Sketch example
- Typical process to enable:
  - Make sure your document reading code is concurrency-safe
  - Override in your NSDocument subclass:

```
+ (BOOL)canConcurrentlyReadDocumentsOfType:(NSString *)type
```

  - Return YES for those types you can concurrently open

NSDocument

- Documents of that type may then be loaded in the background
  - Some NSDocumentController and NSDocument methods will be invoked on nonmain threads
- Don’t do UI in document reading code
  - Only read the data and create the model objects
  - On failure, just return an error as usual
  - Disable undo registration during document reading

Concurrent Drawing

Concurrent Drawing

- Views can be set to indicate that they can be drawn concurrently
  - -setCanDrawConcurrently: YES
  - Subviews do not inherit this property from the superview
- A view with this set may be drawn on a background thread
- This may produce a performance improvement
  - Largest potential win is when there are multiple “slow” views

Concurrent Drawing

- Views not set to be concurrent draw on the main thread
- -display is still done on the main thread and is still synchronous
  - Further activity and event-handling on main thread is blocked
  - So the thread-safety requirements on model don’t change

Enabling Concurrent Drawing

- Measure draw timing
- Find expensive views
- Make those views' drawing concurrency-safe
- Mark them as being able to draw concurrently

- Measure draw timing

Concurrent Drawing Caveats

- Layer-backed views are not drawn concurrently
- Overlapping sibling views are not handled correctly
- Mostly an option for custom views
  - AppKit views and controls are not guaranteed to be concurrent-safe
- See release notes for more information
  - Timing measurement and debugging tips

Wrapping Up

More Information

Michael Jurewitz
Developer Tools and Performance Evangelist
jurewitz@apple.com

Documentation
Concurrency Programming Guide
http://developer.apple.com/wwdc/attendee/sessions
http://developer.apple.com/documentation

AppKit Release Notes
Foundation Release Notes
Available in Xcode

Related Sessions

Designing Your Cocoa Application for Concurrency	Nob Hill Tuesday 3:30PM
Programming with Blocks and Grand Central Dispatch	Marina Wednesday 2:00PM
Objective-C and Garbage Collection Advancements	Russian Hill Wednesday 5:00PM
Understanding Grand Central Dispatch in Depth	Presidio Thursday 9:00AM
Migrating Your Application to Grand Central Dispatch	Pacific Heights Thursday 2:00PM
Cocoa Tips and Tricks	Nob Hill Thursday 5:00PM

Labs

Concurrent Programming in Cocoa Lab	OS X Lab D Thursday 9:00 AM
Cocoa Open Lab	OS X Lab C Thursday 2:00PM

Summary

- Concurrency has been enhanced in some areas of Cocoa
  - New APIs, implementation changes
- NSOperation and NSOperationQueue are fundamental to concurrency in Cocoa
- Queues and operations and blocks are going to be fundamental to the OS

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09